Extending Context Window of Large Language Models via Semantic Compression

Weizhi Fei[†] [‡], **Xueyan Niu**[*] [‡], **Pingyi Zhou**[§], **Lu Hou**[§], **Bo Bai**[‡], **Lei Deng**[‡], **Wei Han**[‡]
[†] Department of Mathematical Sciences, Tsinghua University, Beijing, China
[‡] Theory Lab, 2012 Labs, Huawei Technologies Co., Ltd.
[§] Noah's Ark Lab, 2012 Labs, Huawei Technologies Co., Ltd.

Abstract

Transformer based Large Language Models (LLMs) often impose limitations on the length of the text input to ensure the generation of fluent and relevant responses due to the quadratic complexity. These constraints restrict their applicability in long text scenarios. In this paper, we propose a novel semantic compression method that enables generalization to texts that are 6-8 times longer without incurring significant computational costs or requiring fine-tuning. Our proposed framework draws inspiration from source coding in information theory and employs a pre-trained model to reduce the semantic redundancy of long inputs before passing them to the LLMs for downstream tasks. Experimental results demonstrate that our method effectively extends the context window of LLMs across a range of tasks including question answering, summarization, few-shot learning, and information retrieval. Furthermore, the proposed semantic compression method exhibits consistent fluency in text generation while reducing the associated computational overhead.

1 Introduction

The recent successful release of large language models (LLMs) such as ChatGPT (Radford et al., 2019) and LLaMA (Touvron et al., 2023) have sparked significant research efforts from both industry and academia. These LLMs have demonstrated the ability to engage in fluent and coherent conversations with human users, and have shown exceptional performance across various tasks, including document summarization, question answering, dialogue bots, and code generation.

One critical issue faced by state-of-the-art (SoTA) LLMs is the restriction on the length of text that can be inputted into the model at once. When the input context exceeds the limit of the context window, the performance of these models declines rapidly. This limitation poses a challenge to current LLMs when it comes to handling long texts such as scientific papers, novels, and legal contracts. As a result, there has been a growing interest in finding ways to extend the input length without significantly compromising the models' performance.

The limitation on the context window primarily stems from the quadratic computation of the self-attention mechanism in the transformer. Handling lengthy texts significantly increases the computational costs in terms of both memory and time. Typically, models are trained on shorter contexts, so the maximum sequence length (i.e., the context window) is fixed. If the models are compelled to generate contexts that exceed the context window, they tend to compromise the quality of the output due to the lack of position encoding information that is not learned from the training. Furthermore, generating long sequences imposes substantial memory requirements on the computational device. This accumulation of memory requirements and the lack of effective position encoding can result in length generalization failure (Anil et al., 2022), where the models struggle to generate meaningful and coherent text beyond a certain context window size.

Some techniques have been developed to address the aforementioned challenges. One approach is to devise architectures with nearly linear complexity, which enables efficient scaling to handle very long sequences. However, training a large model from scratch incurs substantial costs. Another strategy involves employing interpolation and fine-tuning techniques to adapt the position encoding to unseen sequence lengths. While this method has the potential to improve the overall performance of LLMs on long sequences, it still demands significant time and GPU resources for fine-tuning and inference. Therefore, methods that do not necessitate altering the parameters of the pre-trained models are more efficient and resource-friendly.

[*]Correspondence to: niuxueyan3@huawei.com.

Findings of the Association for Computational Linguistics: ACL 2024, pages 5169–5181

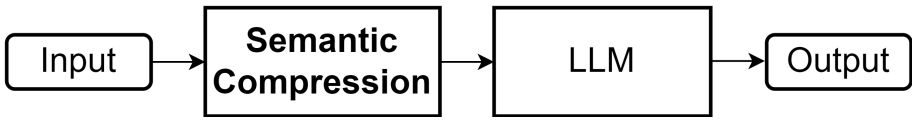

Figure 1: With the inclusion of the semantic compression module, the redundancies in the input are eliminated, thereby effectively extending the context window. The semantic compression is reminiscent of the concept of source coding in information theory.

While most previous algorithms involve modifying the pre-trained models, we instead exploit the statistical properties of the natural language input. One empirical phenomenon, known as Zipf's law (Zipf, 2016), observes that a small set of the most frequent word tokens in a large corpus of natural language account for almost all occurrences. This pattern arises from the tendency of language users to minimize effort in their daily conversations. Consequently, by utilizing an expanded vocabulary, sentences can be significantly shortened while preserving the semantic meaning. Moreover, it is common for language users to include redundant words during communication (Strunk Jr, 2007). These language habits are prevalent among users, and we propose to include a semantic compression module to mitigate the associated redundancies.

Our proposed semantic compression method, reminiscent of lossy source coding in information theory, extends the context window by equivalently shortening the long text input while preserving the semantic meaning. This procedure is conducted before inputting the tokens into the pre-trained LLMs. As illustrated in Fig. 1, the input undergoes compression before being transmitted to the LLM for potential downstream tasks. The semantic compression method can be customized and optimized for downstream tasks, taking into consideration practical constraints such as time and memory resources. The implementation of the semantic compression module is straightforward and can be easily incorporated into other interpolation-based context window extension methods and black box APIs. Our method demonstrates enhanced performance compared to SoTA interpolation-based methods on a range of tasks, including single-document question answering, multi-document question answering, summarization, few-shot learning, and information retrieval, using real-world datasets while incurring no extra parameter updates or memory consumption. Empirically, the proposed method is computationally efficient and achieves 6-8 times context window extension.

Our contributions:

- We propose a context window extension framework for LLMs that serves as a plug-and-play tool to mitigate redundancies in input texts by efficiently performing topic modeling and semantic compression.
- We construct a graph representation of the input to identify distinct sections of the text that pertain to different topics. The result is the segmentation of long texts into separate chunks, each focusing on a specific topic. We then conquer each chunk independently, resulting in a concise version of the original texts. This compression technique helps to condense the information while preserving the key ideas and contexts.
- We demonstrate the applicability of our proposed semantic compression method through extensive experiments. The results highlight the advantages of our method in several key applications, including single-document question answering, multi-document question answering, summarization, few-shot learning, and information retrieval.

2 Related Work

2.1 Efficient Attention Operations

Due to the self-attention mechanism, the inference cost of LLMs grows quadratically with respect to the sequence length. One line of research aims to reduce the complexity when handling long contexts. Dai et al. (2019) present Transformer-XL which utilizes segment-level recurrence agency and a novel positional encoding scheme. Beltagy et al. (2020) introduce Longformer with a sparse attention mechanism that scales linearly with sequence length. Bo (2021) provides a faster transformer, RWKV, which exploits the strength of RNN and has linear complexity during inference. Dao et al. (2022) propose FlashAttention, a chunking strategy for the input, and utilize recomputation to avoid the quadratic complexity of attention computation.

While these methods have the potential to handle longer input sequences (Ding et al., 2023), training new models can be very costly. Moreover, these methods are still not effective when dealing with out-of-distribution content lengths.

2.2 Position Extrapolation and Interpolation

A larger group of research focuses on adapting existing LLMs trained on short texts to accommodate longer text during inference (Anil et al., 2022). The key challenge resides in the position encoding of the input, which has solely been trained on short texts, therefore is inadequate for handling long texts. Current studies are usually based on the relative positional encoding, Rotary Position Embeddings (RoPE) (Su et al., 2024) which is widely adopted due to its strong extrapolation capabilities. Chen et al. (2023b) develop the Position Interpolation (PI) method to linearly scale the positional encoding of long text into trained encoding. Their experiments show that an effective long-context model can be achieved with just 1000 fine-tuning steps. Peng et al. (2023) introduce YaRN, an efficient extrapolation mechanism using the neural tangent kernel which dynamically scales the logits. Chen et al. (2023a) further proposes continuous dynamics and utilizes ordinary differential equations to fit the length scaling factor.

The introduction of new positional embeddings requires fine-tuning long sequences to adapt to the unseen increased length, which can be computationally expensive. To address this problem, LongLoRA is introduced by Chen et al. (2023c), offering an efficient fine-tuning method with limited computational costs. Unlike the above methods, our approach does not require fine-tuning on long texts and is a low-cost length-extending solution.

2.3 Prompting Compression

There are ongoing efforts to extend the context window through smart prompting designs. Wingate et al. (2022) employ soft prompts to compress the context so that the compressed prompts can retain a substantive amount of information. Chevalier et al. (2023) further presents AutoCompressor, which utilizes unsupervised learning to adapt LLMs to compress long contexts into compact summary vectors and then extends the original length of the base model by conditional language modeling. Li (2023) leverage mutual information to eliminate redundant tokens and enhance inference efficiency. Similarly, Jiang et al. (2023) employ a small language model to identify and remove non-essential tokens in the input prompts. These methods collectively showcase the potential of prompt compression in extending the context window of LLM. Both Zhou et al. (2023) and Wang et al. (2023a) recurrently apply LLMs to summarize the input texts to maintain long short-term memory for specific purposes such as story writing and dialogue generation, respectively.

More details on dealing with long text in LLMs are provided in the survey by Huang et al. (2023).

3 Methodology

We now introduce our semantic compression method for extending the context window. The core idea is to compress the input into shorter texts without losing the key information or important details. This enables us to effectively include more content within the fixed input length constrained by the base LLM. Fig. 2 provides an overview of our method, which leverages pre-trained summarization models commonly used in Natural Language Processing (NLP).

Like LLMs, existing summarization methods also have limitations regarding the length of the input. Here, we propose a divide-and-conquer approach that takes into account the topic structure of the text. By identifying distinct topics in the lengthy text and dividing the text into blocks that exhibit a certain level of mutual independence, the content within each block can be compressed efficiently owing to their statistical correlation. The blocks can then be processed in parallel using pre-trained models, and the results are combined to create a condensed textual input that can be processed by the base LLM. This approach provides a computationally efficient way of removing redundancies in long texts which effectively extend the context window of the input.

3.1 Model

Real-world textual content, such as speech and books, frequently displays hierarchical structures, wherein each section is structured around a particular topic, and different sections differ in a topic in a sequential manner. This hierarchical structure, based on topics, bears resemblance to cliques in graphs. To identify this structure within long texts, we utilize weighted graphs to represent them and employ clustering methods to detect cliques in these graphs. The cliques can then be utilized

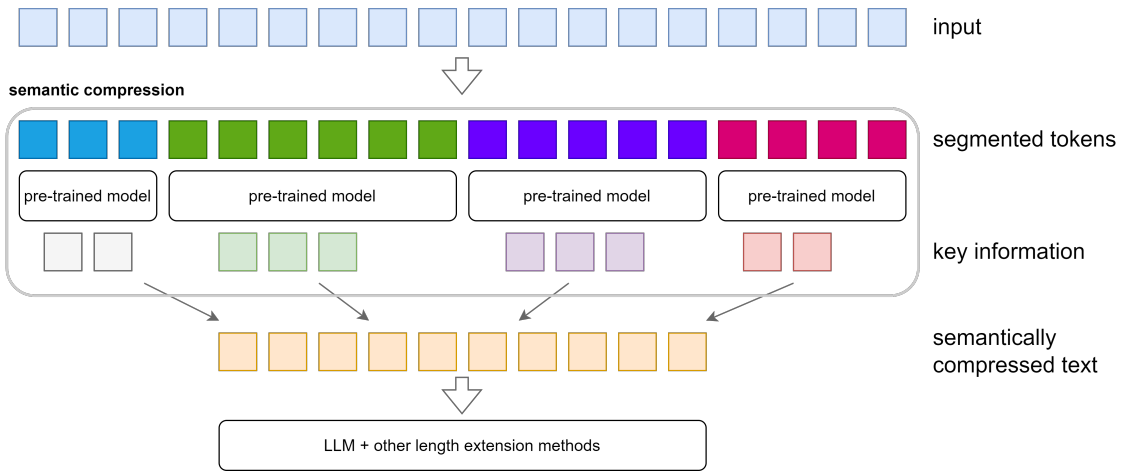

Figure 2: An illustration of our semantic compression method. The input text is initially segmented into topic-based chunks, utilizing the graph representation. Subsequently, these chunks undergo refinement using pre-trained models to ensure the preservation of key information. Finally, the refined chunks are assembled in accordance with the original order. The resulting texts, which have been semantically compressed, are approximately 6-8 times shorter in length compared to the original input. Consequently, they fall within the context window of the LLMs. Furthermore, for an additional extension of the length, other methods such as extrapolation and interpolation-based techniques can be concatenated.

to represent the topic-based content of the text, allowing us to obtain chunks based on the semantic relevance of the topics.

We begin by sequentially constructing sentence-level blocks within given lengths and representing them as nodes in our graph. In this step, we parse the text into different sentences or sub-sentences based on punctuation marks. Next, we sequentially fill the sentence-level blocks until they exceed the desired length before proceeding to the next blocks. Once we have obtained the sentence-level blocks, we build the graph representation of long text $\mathcal{G}$ based on a pre-trained sentence embedding model (e.g., MiniLM (Wang et al., 2020)), where the weight $\mathcal{G}[i][j]$ represents the semantic similarity between the i-th and j-th sentence-level blocks. Typically, this similarity is computed using cosine similarity, which measures the cosine of the angle between two embeddings. If the similarity between two blocks is higher, it indicates that they are closer in topics.

3.2 Topic-Based Chunking

We then apply spectral clustering algorithms [1] on the graph to identify the underlying topic structure. Within each cluster, we group the sentence-level blocks sequentially to obtain the topic-based chunks, which can then be handled simultaneously by the pre-trained model chosen according to the downstream task. The number of clusters can be adjusted to regulate the length of the text following semantic compression. If these semantic chunks still surpass the predetermined length, the identical procedure is repeated to acquire sub-level topic structures.

The obtained topic structures are tree-like, which can be flattened in accordance with the order of the original content. As per the model, each chunk is semantically centered around a specific topic, and these topics are mutually exclusive. Consequently, these chunks can be compressed independently by utilizing a pre-trained summarization model. Choosing from different pre-trained summarization models allows a trade-off between efficiency and effectiveness. Consequently, we can opt to selectively substitute the original chunks with the output of these pre-trained models to ensure the preservation of the underlying topic structure. The semantic compressed text can be forwarded to the LLM directly or in combination with other SoTA extension schemes to further enhance the overall outcome.

3.3 Complexity

To implement our method, two language models may be involved.[2] The first one is the base LLM of interest whose context window is to be extended, and the other serves the purpose of semantic compression, which can be adapted from pre-trained

[1] We present the detailed introduction of our used clustering method in Appendix C.2.

[2] Compared to the language model, the computational overhead of other components is significantly smaller. Please refer to Table 5 for more information regarding this.

summarization models. Since most of the current models are transformer-based, we can assume that their complexities are $k_1 n^2$ and $k_2 n^2$ respectively, where k_1 and k_2 are coefficients that depend on the number of the parameters of the models.

Empirically, the ratio between the summaries generated by the summary model and the original text remains relatively stable. Therefore, we assume a fixed compression coefficient α for the summary model. Given an input text with length L, the origin complexity is $k_1 L^2$. Our method utilizes a divide-and-conquer strategy, dividing the long text into chunks of lengths $l_1, \ldots, l_m$ so that $L = l_1 + \cdots + l_m$. During inference using the compressed context, the complexity is

$$k_1(\sum_{i=1}^{m} \alpha l_i)^2 = k_1(\alpha \sum_{i=1}^{m} l_i)^2 = \alpha^2 k_1 L^2. \quad (1)$$

Let γ_1 and γ_2 be the minimum and maximum input lengths of the summary model, where $\gamma_1 \leq l_i \leq \gamma_2$. Since $m\gamma_1 \leq L$, we have

$$\sum_{i=1}^{m} k_2 l_i^2 \leq \sum_{i=1}^{m} k_2 \gamma_2^2 = m k_2 \gamma_2^2 \leq k_2 \frac{\gamma_2^2}{\gamma_1} L. \quad (2)$$

Thus the complexity of compressing length-L context can be bounded by $k_2 \frac{\gamma_2^2}{\gamma_1} L$, which is linear in L. Furthermore, the summary models usually have smaller parameter sizes, and the entire compression process can be parallelized, allowing for further acceleration. The proposed compression significantly reduces the inference time for long texts. We also present the detailed empirical running time in Appendix D.

4 Experiments

We demonstrate that the proposed method of semantic compression can effectively extend the context window by up to 7-8 times without modifying the parameters of the pre-trained models. Furthermore, the semantic compression module can be seamlessly integrated into existing methods, allowing for further extension of the context window. This versatility enables our approach to be adapted and combined with other techniques such as YaRN, enhancing the overall performance and flexibility. To evaluate the performance of our method, we conduct experiments on several language tasks that require an understanding of long contexts. These tasks include passkey retrieval, single-document question answering, multi-document question answering, summarization, and few-shot learning. In each task, the model is provided with a sequence of context C (typically lengthy texts) and a sequence of text Q (e.g., a prompt), and it is expected to generate the output answer A. Then we conduct the ablation study of the summary model to explore how much parameter work. We conducted an ablation study on the semantic compression module to investigate the parameter size at which the module becomes effective. Additionally, we also investigate the perplexity of generated text by our method in Appendix E.

4.1 Evaluating Tasks

We evaluate our proposed method on various standard benchmark tasks in the long context. Our length extension implementation is based on the pre-trained 7B LLaMA model (Touvron et al., 2023), where the context window size of this model is 4096. The semantic compression module utilizes bart-large-cnn and please refer to Appendix C for further details. We begin with the evaluation of passkey retrieval because this task can present the procedure of our methods. Then we assess the comprehensive ability on four kinds[3] of natural language task from LongBench (Bai et al., 2023). Lastly, we conduct the ablation study of the semantic compression module based on LongBench.

Passkey Retrieval Retrieval has been an important application of LLMs. We evaluate the proposed method using a synthetic task for passkey retrieval introduced by Mohtashami and Jaggi (2023), where prompts are synthesized to conceal a generated passkey within a randomly chosen section of a long document. This task assesses the capacity to extract important information from any position within lengthy contexts. An illustration of the task is shown in Fig. 3. The synthetic long text incorporates the passkey digits, and the task for the LLM is to retrieve these digits from the input text. Further specifics can be found in Appendix B.

General NLP Tasks LongBench (Bai et al., 2023) is a multi-task benchmark designed for long text scenarios, consisting of six distinct

[3]Since our semantic compression module relies on a summarization model, the current implementation is not suitable for the remaining code and synthetic tasks in LongBench. We present the remaining results in Appendix A. Additionally, the summarization results for Llama2 7b are also provided in Appendix A.

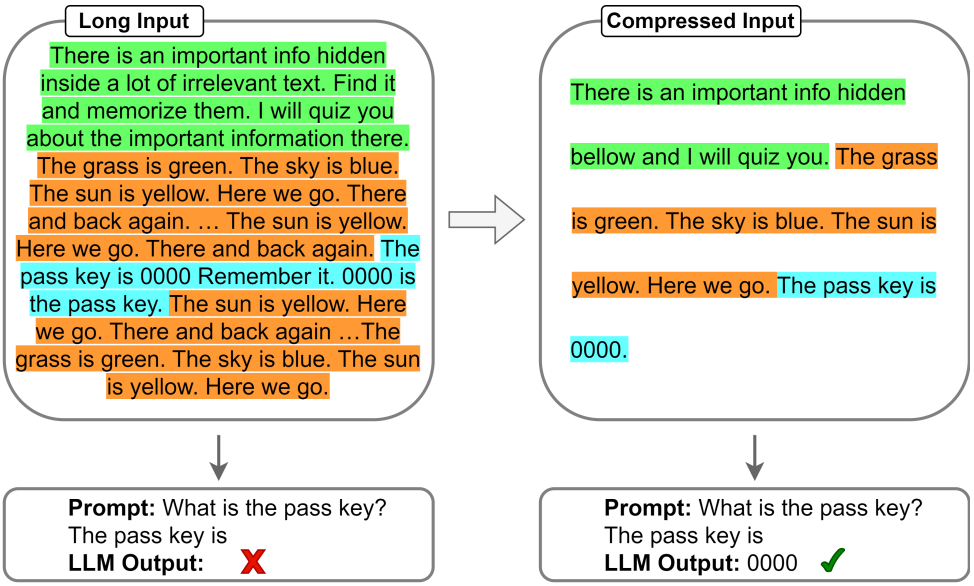

Figure 3: Example of synthetic prompt for the passkey retrieval task (Mohtashami and Jaggi, 2023). The pre-trained LLM is incapable of processing long input due to the context length constraint. By applying semantic compression, the redundant information in the long document is removed, and the compressed input retains essential key information. The LLM can then process the compressed input along with the prompt to generate an accurate answer. Notably, the distinct colors used in the illustration correspond to topic-based chunks.

tasks. In this study, we focus on the three English tasks from the set of four natural language tasks, namely single-document question answering, multi-document question answering, summarization, and few-shot learning. Each of the selected datasets contains 200 instances. Further information can be found in Appendix B.

4.2 Baselines

We choose SoTA solutions from each mainstream approach as our baselines.

Fixed-size chunking To accommodate long context within a fixed-size context window, chunking is a straightforward yet efficient approach. In NLP-related applications, large pieces of text are usually broken down into smaller segments for targeted applications. When the input length exceeds the context window, the fixed-size chunking method (Bai et al., 2023) truncates the input sequence from the middle. This is because the most significant information typically resides at the beginning and end of the sequence.

Interpolation-based method YaRN (Peng et al., 2023) is a computationally efficient method for interpolating position encoding, which dynamically adjusts the Relative Positional Encoding RoPE over dimensions and scales the attention. YaRN offers multiple length-extended models for different versions of Llama2, with the models being trained on a total of 64 GPUs from 8 $\times$ A100 machines. In order to ensure a fair comparison, we choose the model based on Llama2 7B, adjusted from 4k to 64k, as our baseline.

Fine-tuning approach LongLoRA (Chen et al., 2023c) is an efficient approach for fine-tuning that combines LoRA and shifts sparse attention to reduce computational costs. LongLoRA applies this technique to Llama2 models of different sizes, ranging from Llama2 7B, Llama2 13B, to Llama2 70B, with token lengths extended from 4k to 32k on a single 8 $\times$A100 device. In order to ensure a fair and unbiased comparison, we choose the Llama2 7B model with context extension achieved through improved LoRA fine-tuning as our baseline.

5 Results

In this section, we report the results of the above experiments along with a comprehensive analysis.

Passkey Retrieval We present the results of the passkey retrieval task in Fig. 4. We also showcase the compression results of our method for this task in Fig. 3. When employing Llama2 for passkey retrieval, we observe a rapid drop in accuracy to zero once the input length surpasses the window size of 4096. However, by utilizing our method, the retrieval accuracy of the Llama2 model remains above 90% even for inputs with lengths

Task	Dataset (length) \ Method	Long LoRA	Long LoRA (4k)	YaRN	YaRN (4k)	Ours	Ours (4k)	4k
	4k-8k							
Single-Doc QA	NarrativeQA	-	-	-	-	-	-	18.7
	Qasper	11.6	11.8	13.4	12.1	**23.7**	29.6	19.2
	MultiFieldQA-en	24.5	13.2	34.9	32.9	**39.0**	58.7	36.8
Multi-Doc QA	HotpotQA	11.5	8.3	11.3	22.6	**55.8**	50.0	25.4
	2WikiMultihopQA	10.1	10.6	8.9	14.4	**31.7**	61.8	32.8
	MuSiQue	10.0	-	21.1	-	**50.0**	-	9.4
Summarization	GovReport	24.7	28.9	**28.8**	35.0	28.6	32.6	27.3
	QMSum	20.3	17.0	**22.8**	18.7	21.5	22.2	20.8
	MutiNews	0.0	0.0	1.2	18.9	**23.8**	27.3	25.8
Few-Shot Learning	TREC	65.8	54.2	**70.9**	50.0	55.7	54.2	61.5
	TriviaQA	87.6	80.6	**90.9**	88.9	83.6	75.0	77.8
	SAMSum	**43.1**	40.8	40.4	39.9	41.3	43.7	40.7
	8k-16k							
Single-Doc QA	NarrativeQA	9.2	-	13.9	-	**17.3**	-	18.7
	Qasper	-	11.8	10.3	12.1	**20.6**	29.6	19.2
	MultiFieldQA-en	22.5	13.2	18.9	34.4	**35.9**	58.7	36.8
Multi-Doc QA	HotpotQA	8.9	8.3	8.7	22.6	**35.1**	50.0	25.4
	2WikiMultihopQA	9.5	10.6	9.9	14.4	**30.5**	61.8	32.8
	MuSiQue	6.1	-	4.2	-	**24.2**	-	9.4
Summarization	GovReport	24.0	28.9	25.1	35.0	**28.8**	32.6	27.3
	QMSum	22.5	17.0	21.8	18.7	**23.3**	22.2	20.8
	MutiNews	0.0	0.0	0.0	18.9	**22.7**	27.3	25.8
Few-Shot Learning	TREC	**80.4**	54.2	77.3	50.0	66.0	54.2	61.5
	TriviaQA	86.5	80.6	**89.1**	88.9	80.1	75.0	77.8
	SAMSum	**44.5**	40.8	43.8	39.9	38.7	43.7	40.7
	16k-32k							
Single-Doc QA	NarrativeQA	**12.4**	-	8.6	-	11.9	-	18.7
	Qasper	-	11.8	9.2	12.1	**28.3**	29.6	19.2
	MultiFieldQA-en	36.5	13.2	32.6	32.9	**39.0**	58.7	36.8
Multi-Doc QA	HotpotQA	9.3	8.3	10.1	22.6	**26.0**	50.0	25.4
	2WikiMultihopQA	7.9	10.6	10.7	14.4	**31.4**	61.8	32.8
	MuSiQue	5.4	-	5.0	-	**15.1**	-	9.4
Summarization	GovReport	24.7	28.9	**26.4**	35.0	**26.4**	32.6	27.3
	QMSum	20.0	17.0	20.8	18.7	**22.8**	22.2	21.5
	MutiNews	0.3	0.0	0.3	18.9	**21.0**	27.3	26.4
Few-Shot Learning	TREC	-	54.2	-	50.0	-	54.2	61.5
	TriviaQA	88.8	80.6	**90.1**	88.9	81.4	75.0	77.8
	SAMSum	**44.7**	40.8	43.6	39.9	40.6	43.7	40.7
	32k+							
Single-Doc QA	NarrativeQA	oom	-	oom	-	19.4	-	18.7
Summarization	GovReport	oom	28.9	oom	35.0	21.8	32.6	27.3
	QMSum	oom	17.0	oom	18.7	18.8	22.2	21.5

Table 1: Comparison of our method with other baselines on many tasks from the LongBench. Method (4k) denotes evaluation results on texts shorter than 4k. The last column, labeled 4k, showcases the performance of the Llama2-7B-chat-4k baseline.

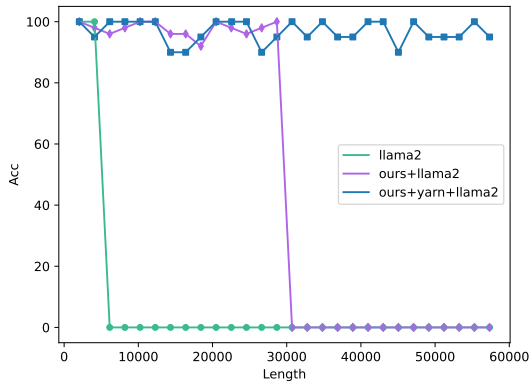

Figure 4: The Accuracy of different model variants on the passkey retrieval task.

of up to 30,000. This indicates that the semantic compression method extends the context window size of the language model by approximately 7-8 times. Furthermore, we combine our method with the SoTA interpolation-based method, YaRN, to further expand the context window size to up to 60,000, while consistently maintaining an accuracy above 90%. This demonstrates that our method can further extend the context window by integrating with current SoTA interpolation approaches.

General NLP Tasks We present our results on various general NLP tasks in Table 1, including single-document question answering, multi-document question answering, summarization, and few-shot learning. When the token length is less than 4k, there is no need to compress the context, and our method performs at the same level as the original Llama2 model. However, both the interpolation-based method YaRN and the fine-tuning approach LongLoRA negatively impact the performance of the Llama2 model across almost all tasks. In the 4k-8k range, our method outperforms others in 6 out of 11 tasks. It is worth noting that our model performs slightly worse in the few-shot learning task. This can be attributed to the fact that few-shot learning necessitates more detailed information, whereas our compression scheme maintains information within a fixed window. Moving on to the 8k-16k range, our method achieves the best results in 9 out of 12 tasks, exhibiting similar performance to the 4k-8k range. In the 16k-32k range, our method outperforms others in 8 out of 11 tasks. In the 32k+ range, other methods fail due to out-of-memory issues, while our method still maintains 70% of the performance achieved in the 4k range.

Table 2: Ablation study the neural compression models. Large represents Facebook bart-large-cnn (406M parameters). Medium represents sshleifer distilbart-cnn-12-6 (320M parameters). Small represents Falconsai text_summarization (60M parameters).

Task	Single	Multi	Sum	Few	Mean
Large	28.75	21.45	24.77	50.27	31.31
Medium	26.59	22.60	24.83	49.03	30.76
Small	27.63	20.92	24.40	50.00	30.74

Ablation Study The summary model in the semantic compression module plays a crucial role. The larger the parameter size of the model, the better its ability to generalize, and the same principle applies to the task of summarization as well. In this section, we investigate the effect of parameter size on the performance of the semantic compression module. We adopt three models with parameter sizes of 406M for bart-large-cnn, 320M for distilbart-cnn-12-6, and 60M for text_summarization, which we refer to as "Large", "Medium", and "Small", respectively.

By only changing the summary model, we evaluate their performance on the English tasks in LongBench dataset. The results in Table 2 demonstrate that as the size of the summary model decreases, the average scores of the compression module built by the three summary models slowly decrease. Though the "large" did not achieve the best performance in the summarization task, it exceeded the average scores in the other three tasks. Even with the small model, the performance remains competitive. The above discussion indicates the robustness and reliability of our approach.

6 Conclusion

In this work, we propose a novel approach to addressing the limitation of context window length in large language models using semantic compression. By leveraging the statistical properties of natural language and exploiting redundancy in communication, we are able to significantly shorten texts while preserving their semantic meaning. This allows for a 6-8 time extension of the context window without the need for modifying the parameters of the pre-trained model or incurring additional computational costs. The experiments demonstrate that our proposed semantic compression method outperforms current SoTA approaches on the multi-task long context understanding benchmark, Long-

Bench. Furthermore, the implementation of our semantic compression module is straightforward and can be easily integrated into other interpolation-based methods and black box APIs. This provides flexibility and adaptability to different downstream tasks, considering practical constraints such as time and memory resources. We believe our work can lead to a simpler context window extension method to be used in practice, thereby reducing the cost of large language models.

7 Limitation

The semantic compression method we propose is a lossy compression that retains only key semantic information. When faced with fine-grained tasks such as counting the occurrences of a specific name in the full text, our method may struggle to provide accurate answers. The current implementation just supports English since there are many effective summary models. We will introduce the multilingual version in the future.

8 Potential Risks

As a context window extension approach for large language models, our proposed semantic compression approach enables the model to process long texts that are 6-8 times the length of the original context window. The experiments and evaluations we conducted utilized publicly available academic datasets, thus avoiding direct ethical concerns. However, it is worth noting that our method could potentially be used to extract inferred privacy information from long texts in commercial settings. This is a common ethical concern associated with long text models.

9 Future Work

The core assumption of our compression approach is that long text remains redundant (Li, 2023). An apparent direction involves a task-agnostic compression strategy that considers the semantic information from task instructions. The evaluated tasks primarily focus on information retrieval from long texts. Reasoning is a fundamental ability of artificial intelligence systems (Wang et al., 2021; Yin et al., 2024b; Wang et al., 2024), and exploring performance on long text reasoning is well-deserved. The evaluation is based on automatic metrics in NLP, which may be biased. Therefore, it could be beneficial to conduct human evaluations through a crowdsourcing platform or use GPT-4 scores (Sottana et al., 2023).

References

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. 2022. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

PENG Bo. 2021. Blinkdl/rwkv-lm: 0.01.

Guanzheng Chen, Xin Li, Zaiqiao Meng, Shangsong Liang, and Lidong Bing. 2023a. Clex: Continuous length extrapolation for large language models. *arXiv preprint arXiv:2310.16450*.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023b. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023c. Longlora: Efficient fine-tuning of long-context large language models. *arXiv*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *ArXiv*, abs/2305.14788.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, and Furu Wei. 2023. Longnet: Scaling transformers to 1,000,000,000 tokens. In *Proceedings of the 10th International Conference on Learning Representations*.

Alexander Richard Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084.

Weizhi Fei, Zihao Wang, Hang Yin, Yang Duan, Hanghang Tong, and Yangqiu Song. 2024. Soft Reasoning on Uncertain Knowledge Graphs. ArXiv:2403.01508 [cs].

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Yunpeng Huang, Jingwei Xu, Zixu Jiang, Junyu Lai, Zenan Li, Yuan Yao, Taolue Chen, Lijuan Yang, Zhou Xin, and Xiaoxing Ma. 2023. Advancing transformer architecture in long-context large language models: A comprehensive survey.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376. Association for Computational Linguistics.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Yucheng Li. 2023. Unlocking context constraints of llms: Enhancing context efficiency of llms with self-information-based content filtering.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Andrea Sottana, Bin Liang, Kai Zou, and Zheng Yuan. 2023. Evaluation metrics in the era of gpt-4: reliably evaluating large language models on sequence to sequence tasks. *arXiv preprint arXiv:2310.13800*.

William Strunk Jr. 2007. *The Elements of Style*. Penguin.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. 2024. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? *arXiv preprint arXiv:2402.18272*.

Qingyue Wang, Liang Ding, Yanan Cao, Zhiliang Tian, Shi Wang, Dacheng Tao, and Li Guo. 2023a. Recursively summarizing enables long-term dialogue memory in large language models. *arXiv preprint arXiv:2308.15022*.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788.

Zihao Wang, Weizhi Fei, Hang Yin, Yangqiu Song, Ginny Y Wong, and Simon See. 2023b. Wasserstein-Fisher-Rao Embedding: Logical Query Embeddings with Local Comparison and Global Transport. *arXiv preprint arXiv:2305.04034*.

Zihao Wang, Hang Yin, and Yangqiu Song. 2021. Benchmarking the Combinatorial Generalizability of Complex Query Answering on Knowledge Graphs. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1.

David Wingate, Mohammad Shoeybi, and Taylor Sorensen. 2022. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5621–5634, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hang Yin, Zihao Wang, Weizhi Fei, and Yangqiu Song. 2023. $\text{EFO}_k - \text{CQA}$: Towards knowledge graph complex query answering beyond set operation. *arXiv preprint arXiv:2307.13701*.

Hang Yin, Zihao Wang, and Yangqiu Song. 2024a. Meta Operator for Complex Query Answering on Knowledge Graphs. ArXiv:2403.10110 [cs].

Hang Yin, Zihao Wang, and Yangqiu Song. 2024b. Rethinking complex queries on knowledge graphs with neural link predictors. In *The Twelfth International Conference on Learning Representations*.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Peng Cui, Tiannan Wang, Zhenxin Xiao, Yifan Hou, Ryan Cotterell, and Mrinmaya Sachan. 2023. Recurrentgpt: Interactive generation of (arbitrarily) long text.

George Kingsley Zipf. 2016. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Ravenio Books.

A Other results

We present the results of all the tasks on LongBench dataset in Table 3. Compared with the baselines, it can be observed that our method performs well in Single-Doc QA, Multi-Doc QA, and Summarization tasks. We also present the performance of our method using prompting LLM as a neural compressor in Table 4. The summary prompts are the following. Although letting the model perform the summarization task itself can reduce the need for a separate summarization model, it generally results in average performance while increasing complexity.

```
Write a concise summary of the

following text. '''{text}'''SUMMARY:
```

B Datasets

Single-Doc QA assesses the model's ability to retrieve local information, while the summarization task requires the model to know the full text. Multi-Doc QA involves answering multi-hop questions (Wang et al., 2023b; Yin et al., 2024a) across long document content, which poses a significant challenge as it requires sophisticated reasoning abilities (Fei et al., 2024; Yin et al., 2023; Wang et al., 2021). However, there is still room for improvement in Few-shot Learning, Code Completion, and Synthetic tasks.

Single-Doc QA

- **NarrativeQA** (Kočiskỳ et al., 2018) is a standard question-answering dataset that includes books from Project Gutenberg3 and movie screenplays from a list of websites. Question-answer pairs were provided by annotators, so that each of the 1,567 books and scripts has about 30 questions and answers, and two reference answers are given for each question.

- **Qasper** (Dasigi et al., 2021) is a question-answering dataset of NLP publications containing abstractive, extractive, and yes/no questions.

- **MultiFieldQA-en** (Bai et al., 2023) is a dataset created from multiple sources including legal documents, government reports, encyclopedias, and academic publications. Doctoral students were requested to annotate each article's queries and responses.

Multi-Doc QA

- **HotpotQA** (Yang et al., 2018) includes many 2-hop questions written by native speakers based on two related paragraphs.

- **2WikiMultihopQA** (Ho et al., 2020) involves up-to 5-hop questions systematacially constructed by manual templates. Answering these questions requires reasoning paths and can not be solved by local content.

- **MuSiQue**(Trivedi et al., 2022) consists of up to 4-hop questions and removes shortcuts and naturalness questions. Each question contains 2-4 supplement paragraphs which present the reasoning path and related paragraphs.

Summarization

- **GovReport** (Huang et al., 2021) collects detailed reports containing human-written summaries from the U.S. Government Accountability Office and Congressional Research Service. These reports span a wide variety of national policy issues.

- **QMSum** (Zhong et al., 2021) contains annotated meeting-summary pairs across many domains including including product, academic, and committee meetings.

- **MultiNews**(Fabbri et al., 2019) is a multi-document summarization dataset. (Bai et al., 2023) cluster 2-10 news articles discussing the same event or topic, each paired with a human-written summary and form a new long text summarization task.

Table 3: The results of the entire LongBench tasks. The results of our are based on the Facebook bart-large-cnn. We assign the labels 1-1, 1-2, and 1-3 to NarrativeQA, Qasper, and MultiFieldQA-en, respectively. We designate HotpotQA, 2WikiMultihopQA, and MuSiQue as 2-1, 2-2, and 2-3, respectively. Similarly, we assign the labels 3-1, 3-2, and 3-3 to GovReport, QMSum, and MutiNews, respectively. TREC, TriviaQA, and SAMSum are labeled as 4-1, 4-2, and 4-3, respectively. LCC and Repobench-P are denoted as 5-1 and 5-2, while Passage Count and PassageRetrieval-en are labeled as 6-1 and 6-2, respectively.

Tasks	Single-Doc QA			Multi			Sum			Few-shot			Code		Syntax	
Models	1-1	1-2	1-3	2-1	2-2	2-3	3-1	3-2	3-3	4-1	4-2	4-3	5-1	5-2	6-1	6-2
Llama	18.7	19.2	36.8	25.4	32.8	9.4	27.3	20.8	25.8	61.5	77.8	40.7	52.4	43.8	2.1	9.8
yarn	12.4	12.8	27.0	9.9	9.8	5.0	26.8	21.5	13.9	71.5	89.7	42.5	64.8	60.8	2.9	5.5
Our	17.9	24.3	39.6	32.0	33.4	16.0	28.3	21.7	26.3	60.5	80.8	40.3	52.7	51.5	1.0	9.5

Table 4: The results of our(prompt) are based on prompting LLM as a summarized model. The results of our are based on the Facebook bart-large-cnn. We assign the labels 1-1, 1-2, and 1-3 to NarrativeQA, Qasper, and MultiFieldQA-en, respectively. We designate HotpotQA, 2WikiMultihopQA, and MuSiQue as 2-1, 2-2, and 2-3, respectively. Similarly, we assign the labels 3-1, 3-2, and 3-3 to GovReport, QMSum, and MutiNews, respectively. TREC, TriviaQA, and SAMSum are labeled as 4-1, 4-2, and 4-3, respectively.

Tasks	Single-Doc QA			Multi			Sum			Few-shot		
Models	1-1	1-2	1-3	2-1	2-2	2-3	3-1	3-2	3-3	4-1	4-2	4-3
Llama	18.7	19.2	36.8	25.4	32.8	9.4	27.3	20.8	25.8	61.5	77.8	40.7
yarn	12.4	12.8	27.0	9.9	9.8	5.0	26.8	21.5	13.9	71.5	89.7	42.5
Our	17.9	24.3	39.6	32.0	33.4	16.0	28.3	21.7	26.3	60.5	80.8	40.3
Our(prompt)	15.7	19.4	36.7	31.2	33.6	14.9	27.7	20.8	26.6	52.0	78.3	38.4

Few-Shot Learning To construct few-shot learning with long text, (Bai et al., 2023) select a range of training examples in the following datasets to concatenate the context in LongBench.

- **TREC** (Li and Roth, 2002) is a classification dataset with fine-grained class label.
- **TriviaQA** (Zhong et al., 2021) is a classification dataset and involves messenger-like conversations with human-written summaries.
- **SAMSum** (Fabbri et al., 2019) reading comprehension dataset and consists of question-answer pairs annotated with evidence passages.

Passkey The randomly generated prompts of the passkey retrieval task are in the format of Fig. 5.

The datasets we discussed are openly available for academic purposes and are licensed under CC-BY 4.0. Our proposed method is also licensed under CC-BY 4.0 and can be used for academic purposes. However, it is important to note that derivatives of data accessed for research purposes should not be deployed in the real world as anything other than a research prototype, especially for commercial purposes.

C Implementation Details

In this section, we provide details of our algorithm implementation. Our experiments can be conducted on a single V100 GPU with 32GB of memory. Each task on LongBench takes approximately one hour to complete.

C.1 Open-source Models

Our algorithm utilizes several mature open-source models. For graph representation, we make use of the sentence similarity models all-MiniLM-L6-v2 provided by the Sentence Transformer platform, which can be found at the following link: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. For semantic compression, we employ the pre-trained model bart-cnn-12-6[4]. In most of our experiments, we utilize Llama2-7B-chat-4k as the base large language model (Touvron et al., 2023).

C.2 Clustering Algorithms

In our clustering implementation, we utilize Spectral Clustering as a standard method for graph clustering. For example, when there are two clusters, Spectral Clustering solves a convex relaxation of the normalized cuts problem on the similarity graph. It achieves this by dividing the graph into two parts in such a way that the weight of the cut edges is minimized relative to the weights of the edges within each cluster. You can refer to https://scikit-learn.org/stable/modules/clustering.html#spectral-clustering for more details. This approach aligns closely with our objective of identifying topic structures. Furthermore, this method is both simple and effective, as it only requires one parameter: the cluster number. This parameter allows us to control the length of topic-based chunks.

D Running Time

We present the running time analysis of three datasets in Multi-Doc QA in Table 5. The average word lengths of 2wikimqa,

[4]Available at: https://huggingface.co/facebook/bart-large-cnn

There is an important info hidden inside a lot of irrelevant text. Find it and memorize them. I will quiz you about the important information there. The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. (Repeat X Times) The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again. The pass key is 0000 Remember it. 0000 is the pass key. The sun is yellow. Here we go. There and back again (Repeat X Times) The grass is green. The sky is blue. The sun is yellow. Here we go. What is the pass key? The pass key is

Figure 5: The query prompt contains task descriptions, redundant information, passkey information, redundant information, and query information. The passkey information is randomly placed within the text, while the remaining space up to a specified length is filled with redundant information.

hotpotqa, and musique are 4,887, 9,151, and 11,214, respectively. These lengths represent word counts, not token counts, and are sourced from LongBench. However, it is important to note that due to the limitation of a single GPU, we can only process sequences up to 8k in length, which makes direct inference with yarn more advantageous. The results in the table indicate that our proposed approach has a longer inference time compared to direct inference, with the main time consumption occurring during the compression process. Nevertheless, the compression step can be parallelized, leaving room for engineering optimizations. It can be observed that the runtime of the clustering algorithm is relatively minimal compared to other modules, thus we did not delve into detailed discussions on it.

Time (s)	2wikimqa	hotpotqa	musique
AVG.(cluster)	0.02	0.04	0.10
AVG.(compress)	4.73	8.53	15.00
AVG.(our)	6.95	12.39	21.75
AVG.(yarn)	5.85	7.05	7.35

Table 5: The average running time for different datasets. AVG.(yarn) represents the running time for the yarn method on 8k examples. AVG.(our) denotes the running time for our method across all examples. AVG.(cluster) and AVG.(compress) indicate the average time for the clustering and compression components of our method, respectively.

E Fluency

We evaluate the fluency of our semantic compression method using the perplexity score, which is defined as the exponential of the average negative log-likelihood of the probabilistic model P on the distribution D, i.e.,

$$\mathrm{PPL}(D, P) := \exp(-\mathbb{E}_{x \in D} \log P(x)).$$

A smaller perplexity score indicates more fluent sequences that are consistent with the model. We utilize the Llama2 model as our baseline to evaluate the fluency of generated texts by calculating the perplexity (PPL) score. Samples from the GovReport dataset are selected at varying lengths, and the reference texts are compared to the generated texts during

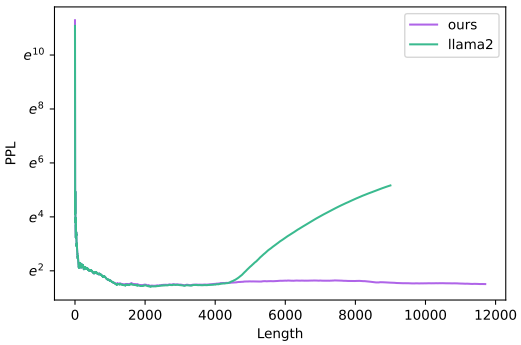

Figure 6: Perplexity on the GovReport dataset was evaluated at different sequence lengths. The perplexity curves of Llama2 (green) and our method (purple) exhibit similar trends for sequences up to 4k in length. However, as the sequence length exceeds the training length of 4k, our method effectively flattens the perplexity curve, indicating that fluency is preserved for longer sequences.

the computation. In cases where the length of the input text exceeds the context window of Llama2, our semantic compression module shortens the input, thereby allowing the model to continue generating new content fluently. The resulting scores are depicted in Fig. 6. The plots indicate that the perplexity of Llama2 initially decreases, but once it surpasses the window length, it rapidly increases. However, when our semantic compression method is employed, the PPL remains consistently low. This suggests that our approach successfully extends the context window up to three times without compromising the generation quality of the language model.

(Peng et al., 2023), which measures the model's ability to predict the text and serves as an indicator of the fluency of the generated output. This analysis allows us to assess not only the effectiveness but also the quality of the generated output.